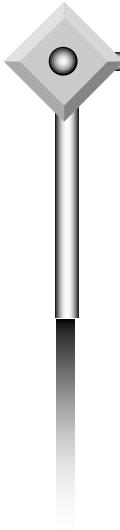




CLARAty: Towards Standardized Abstractions and Interfaces for Robotics Systems

Coupled Layer Architecture for Robotic Autonomy



Issa A.D. Nesnas

Jet Propulsion Laboratory,
California Institute of Technology

In Collaboration with

Ames Research Center
Carnegie Mellon University
University of Minnesota

ICRA 2005 – Principle and Practice of Software Development in Robotics
April 18-22, 2005



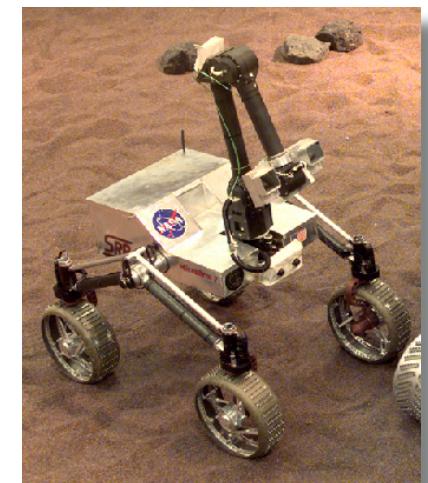
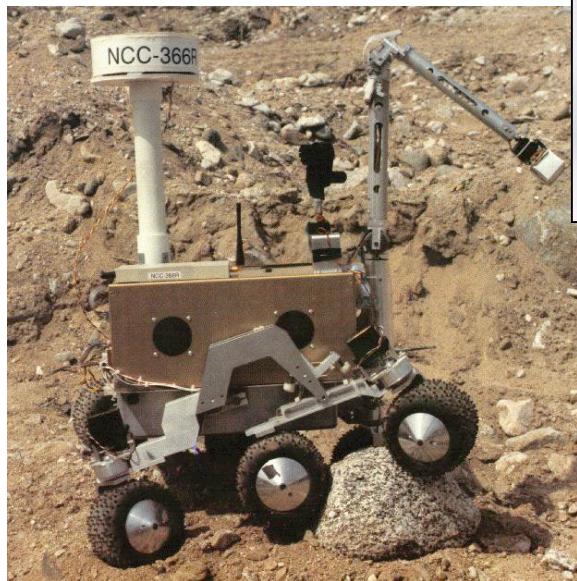
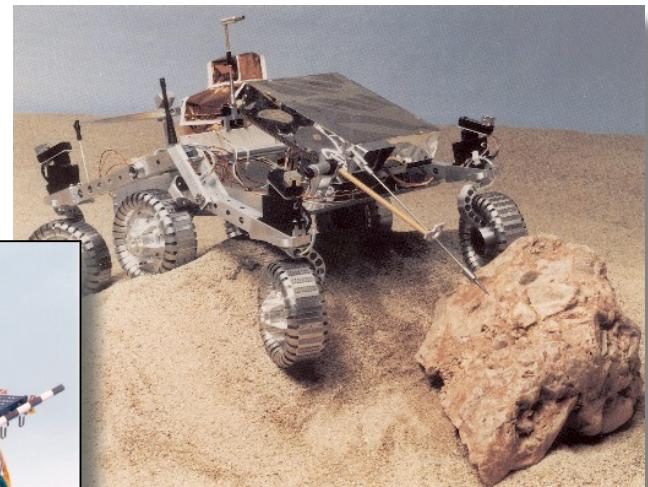
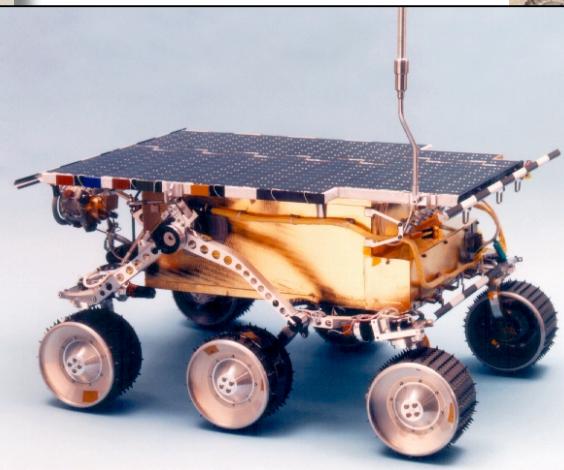
Motivation





Various Rovers Developed by JPL/NASA

JPL





With Different Mobility Mechanisms

JPL





Problem and Approach

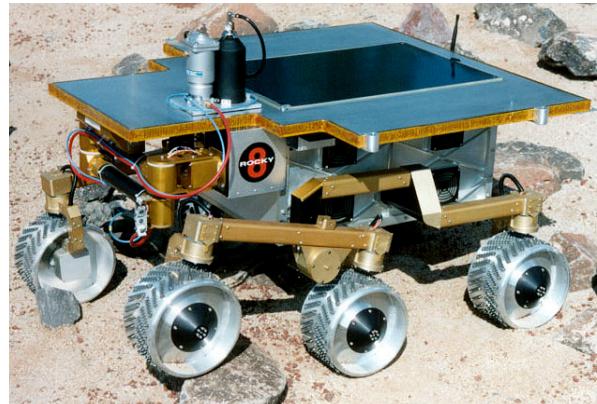


- Problem:
 - Difficult to share software across systems
 - Different hardware/software infrastructure
 - No standard protocols and APIs
 - No flexible code base of robotic capabilities
- Approach
 - Unified robotic framework
 - Capture and integrate legacy algorithms
 - Enable faster technology development
 - Operate heterogeneous robots



JPL

Would like to support ...



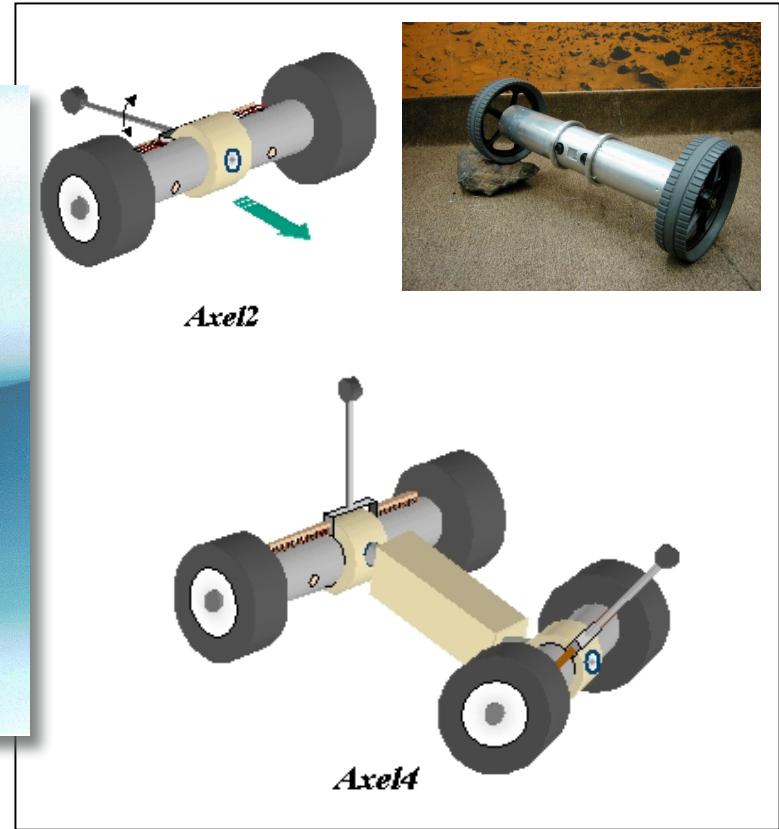
Custom Rovers



Manipulators



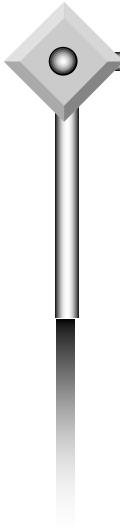
COTS Systems



Reconfigurable Robots



Challenges in Interoperability



- Mechanisms and Sensors
- Hardware Architecture
- Modular and Reusable Software Components

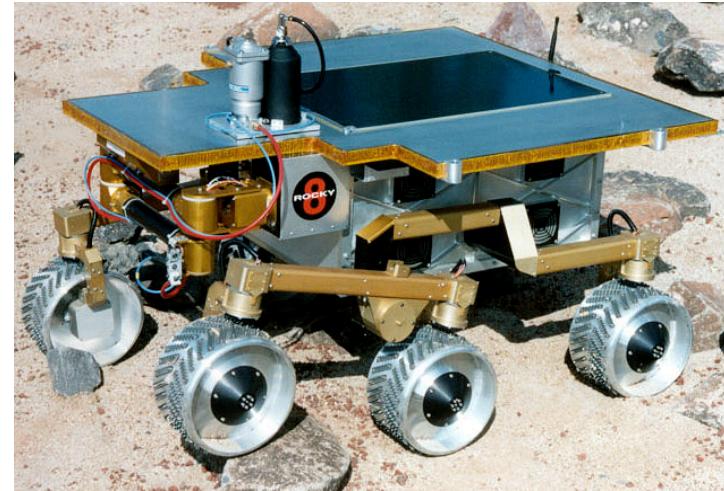


Locomotion

JPL



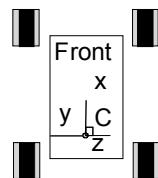
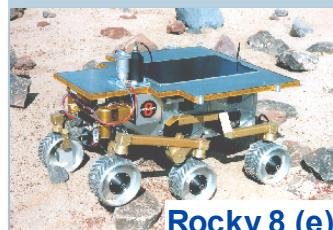
Rocky 7



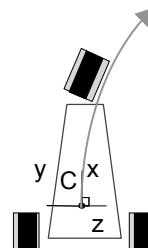
Rocky 8



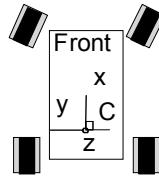
General flat terrain algorithms and specialized full DOF algorithms



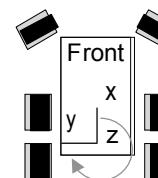
(a)
Skid Steering
(no steering wheels)



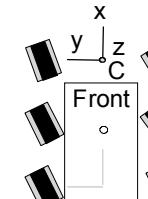
(b)
Tricycle
(one steering wheel)



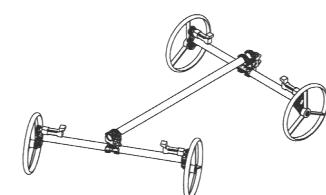
(c)
Two -wheel steering



(d)
Partially Steerable
(e.g. Sojourner,
Rocky 7)



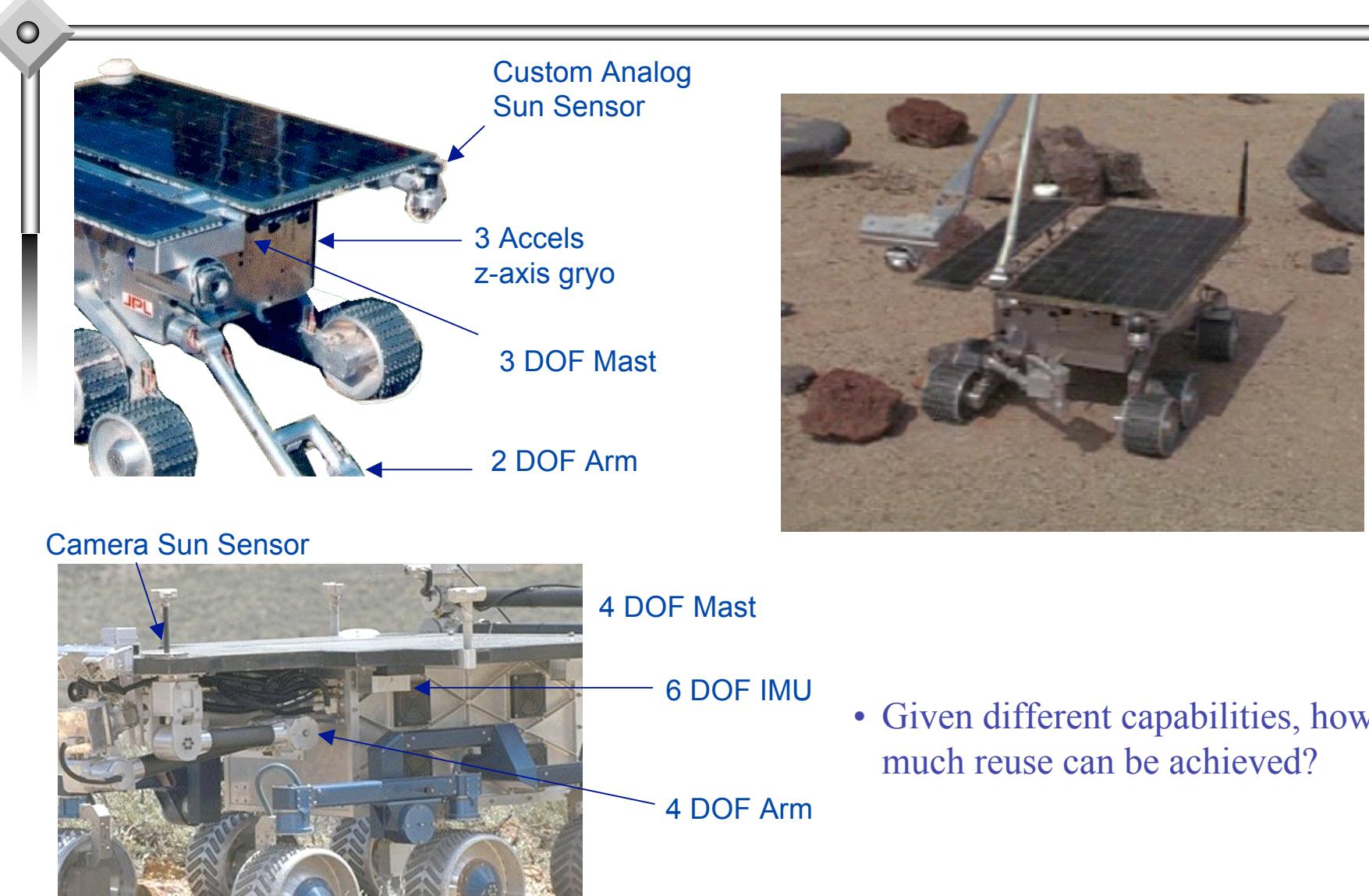
(e)
All wheel steering
(e.g. MER, Rocky8,
Fido, K9)



(f)
Steerable Axle
(e.g.Hyperion)

• • •

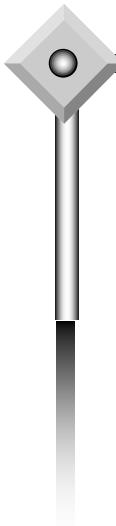
Manipulators and Sensor Suites



- Given different capabilities, how much reuse can be achieved?



Challenges in Interoperability

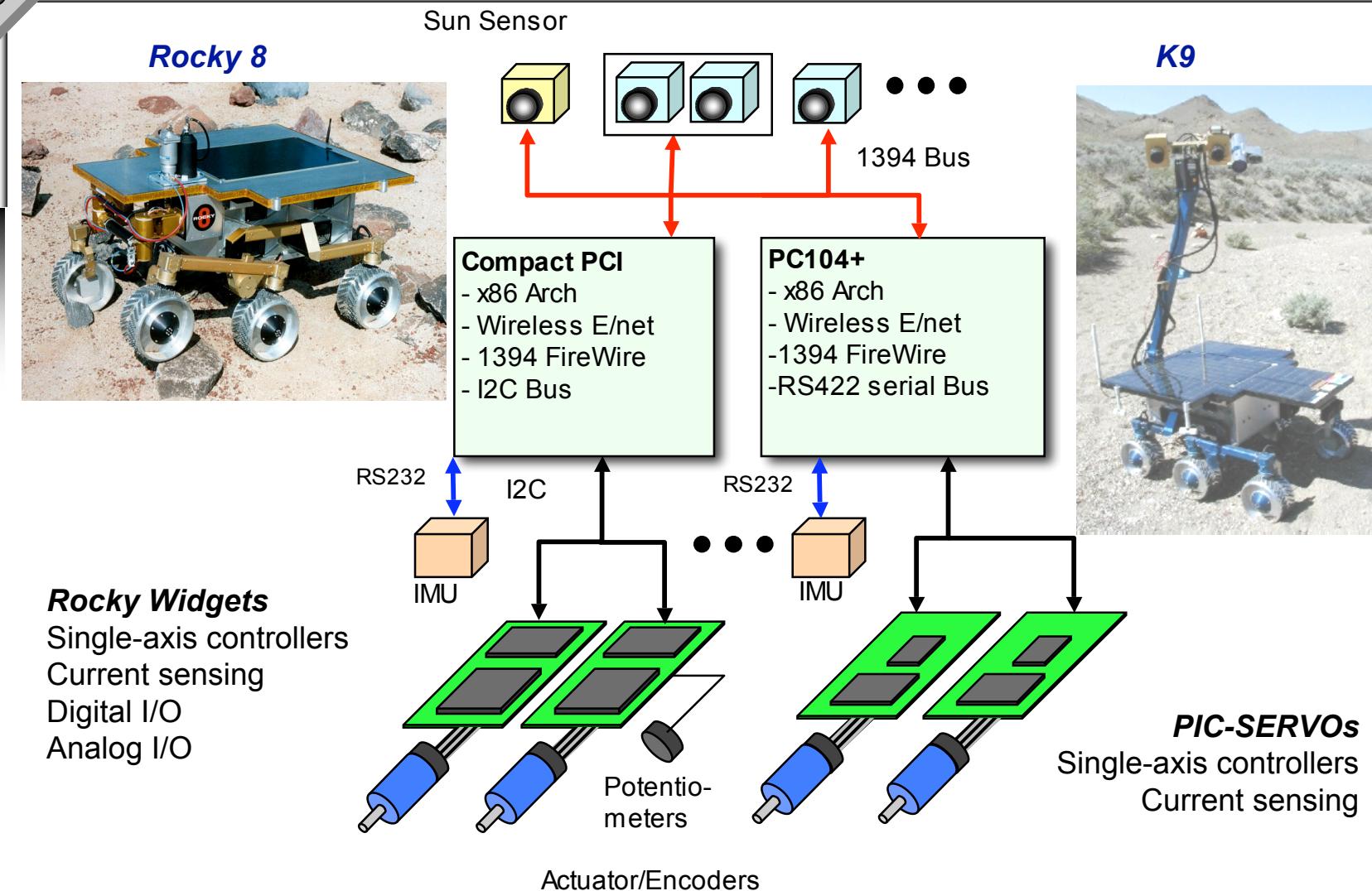


- Mechanisms and Sensors
- **Hardware Architecture**
- Modular and Reusable Software Components



Distributed Hardware Architecture

JPL

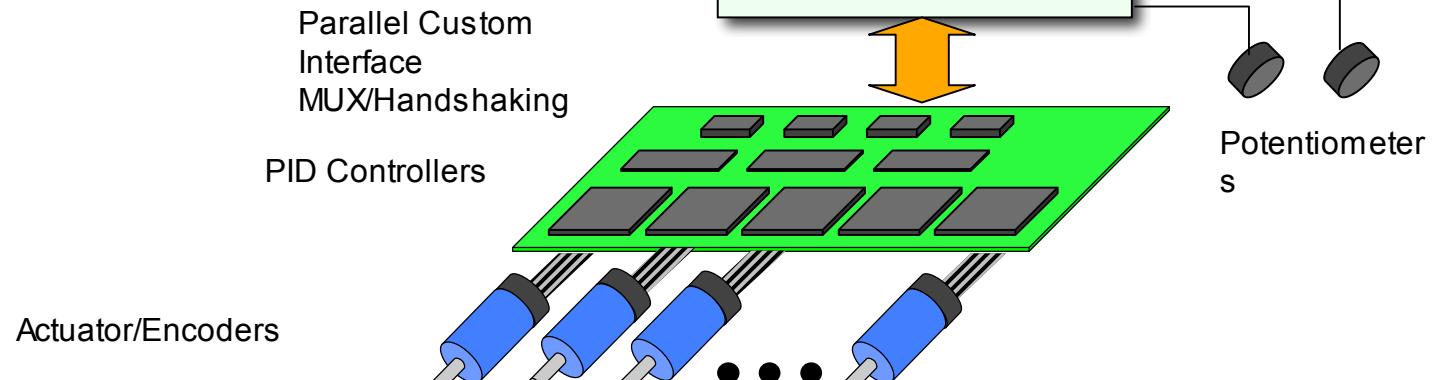




Custom Architecture/Variability



Rocky 7





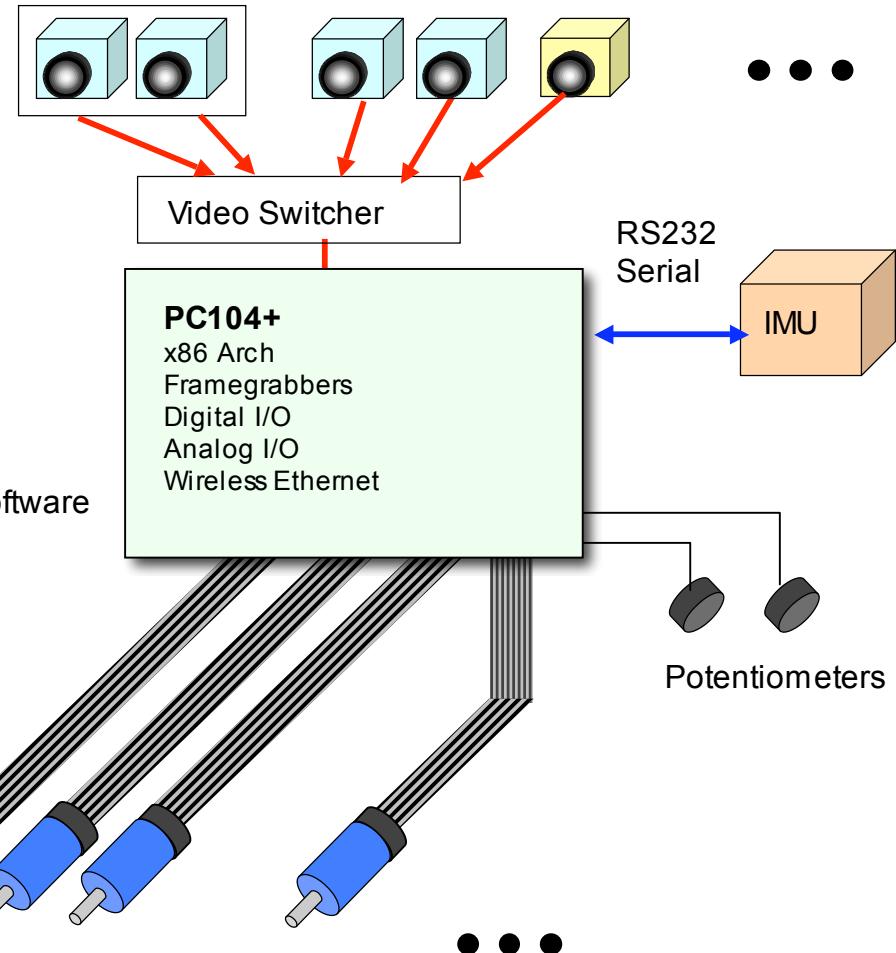
Centralized Hardware Mapped Architecture

JPL



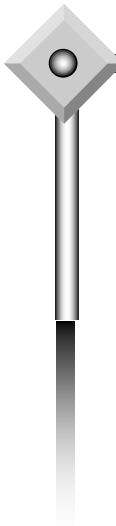
Fido

PID Control in Software





Challenges in Interoperability

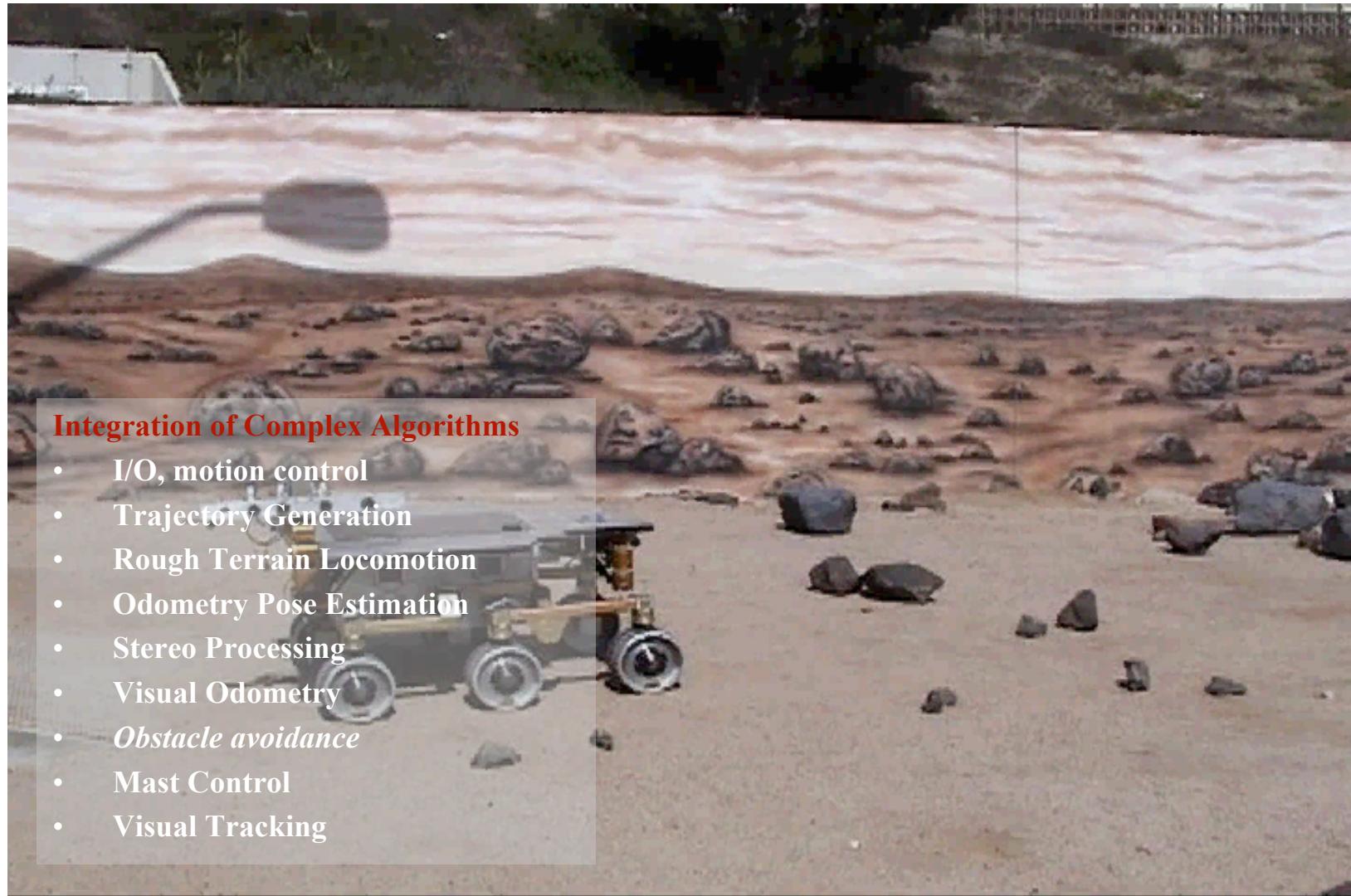


- Mechanisms and Sensors
- Hardware Architecture
- Modular and Reusable Software Components



Designated Target Tracking for Single-Cycle Instrument Placement

JPL





Navigation with Path Planning on Two Rovers

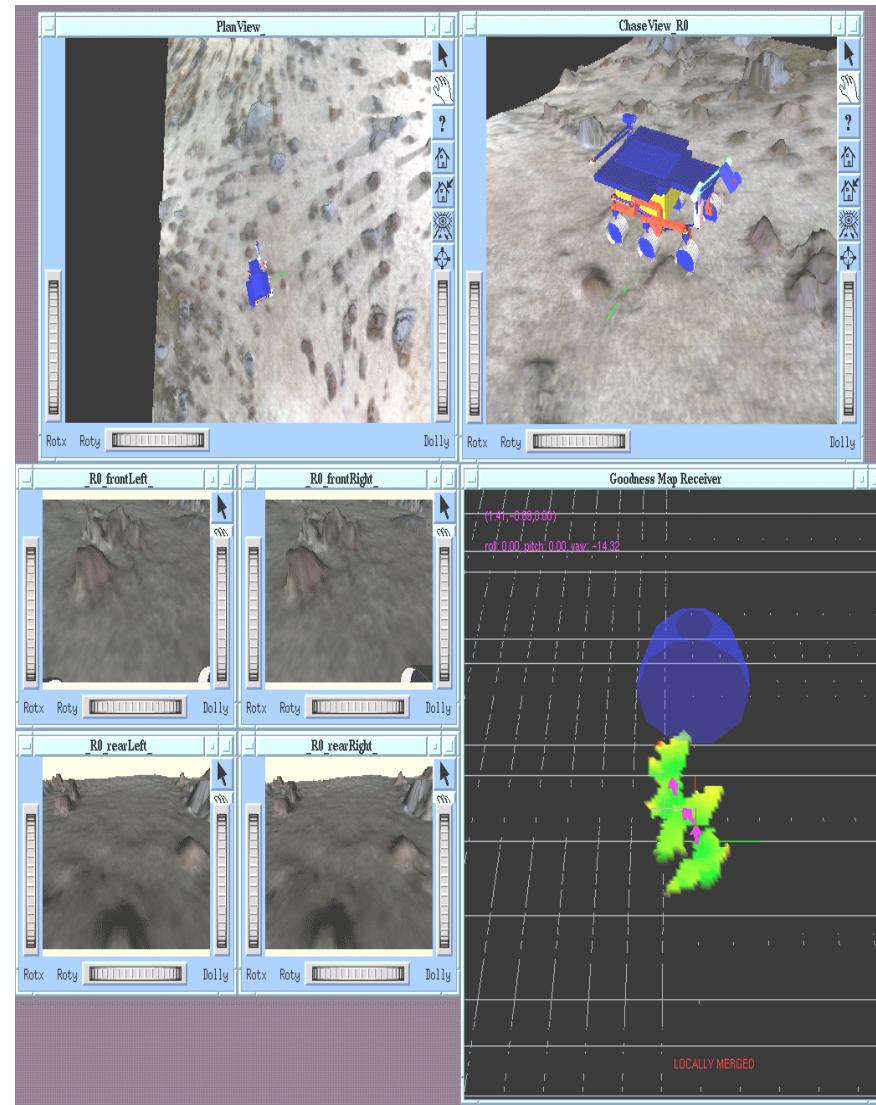
JPL





And with a Simulated Rover

JPL



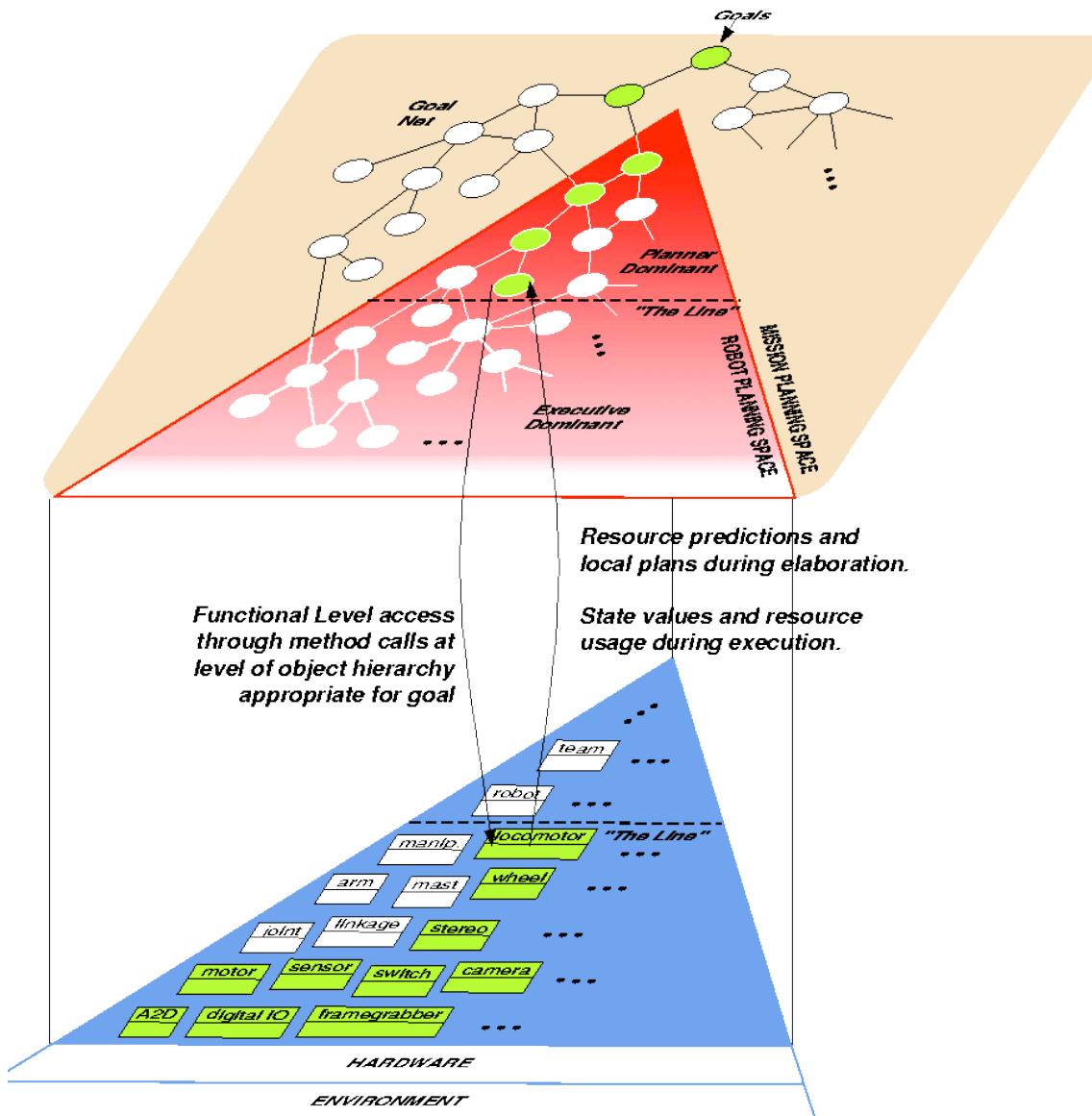


Technical Approach



A Two-Layered Architecture

CLARAty = Coupled Layer Architecture for Robotic Autonomy



THE DECISION LAYER:

Declarative model-based
Global planning

INTERFACE:

Access to various levels
Commanding and updates

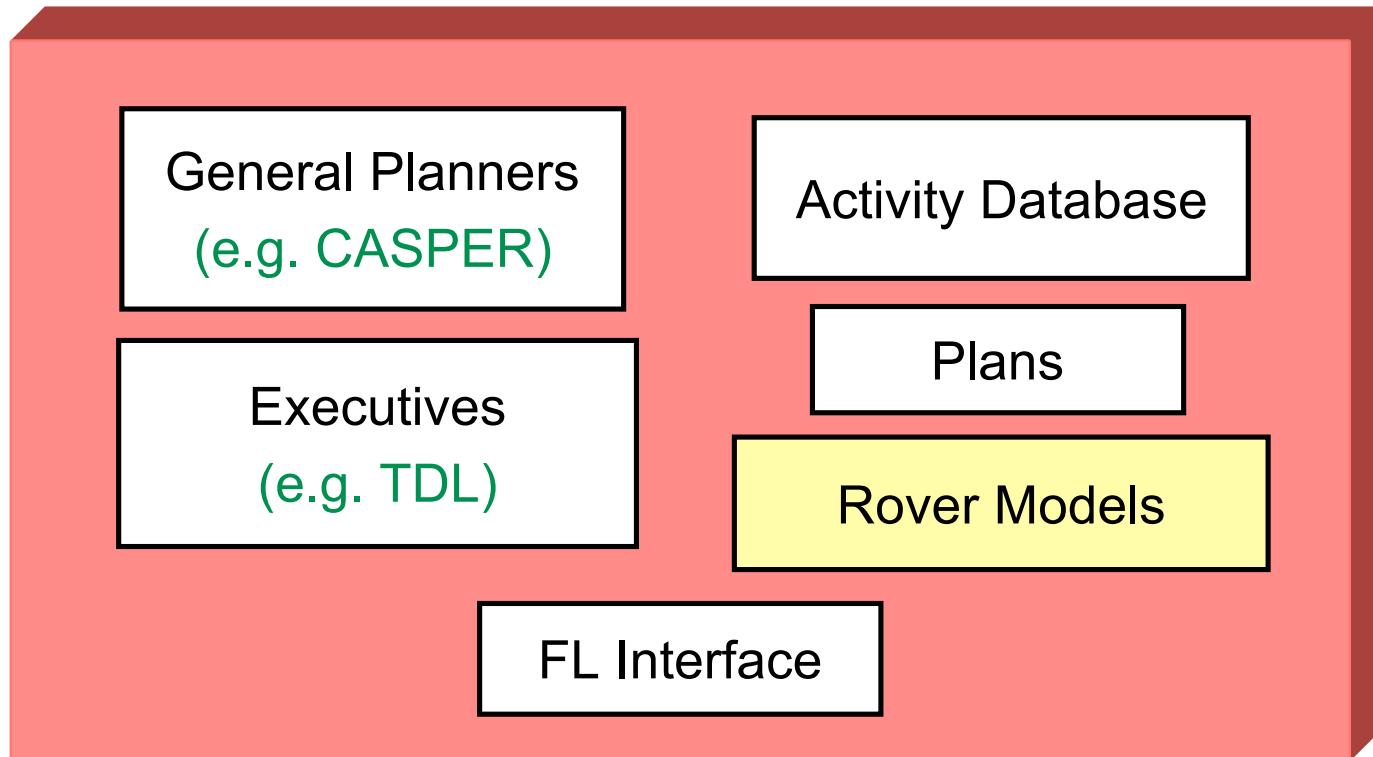
THE FUNCTIONAL LAYER:

Object-oriented abstractions
Autonomous behavior
Basic system functionality

Adaptation to a system

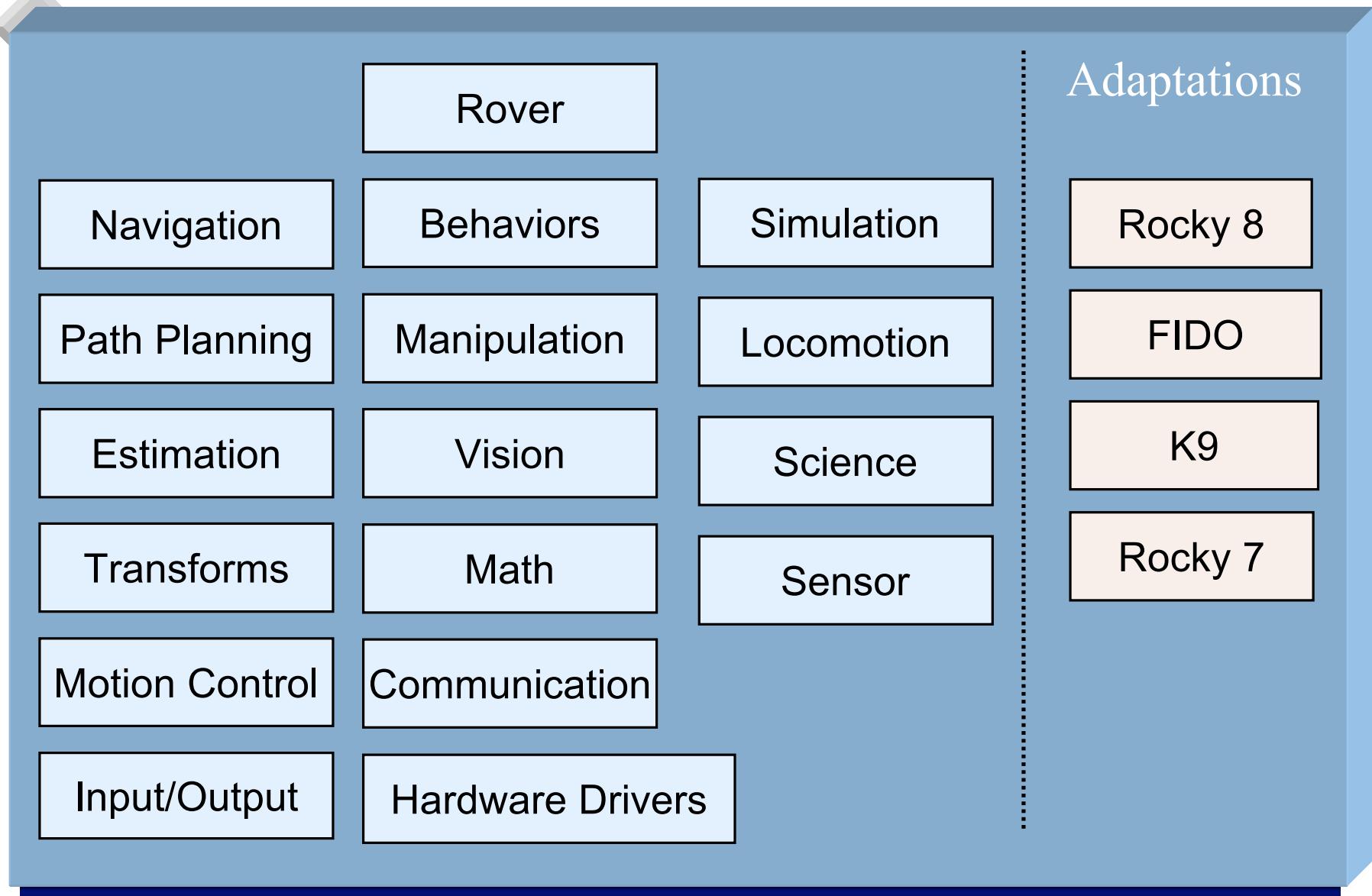


The Decision Layer



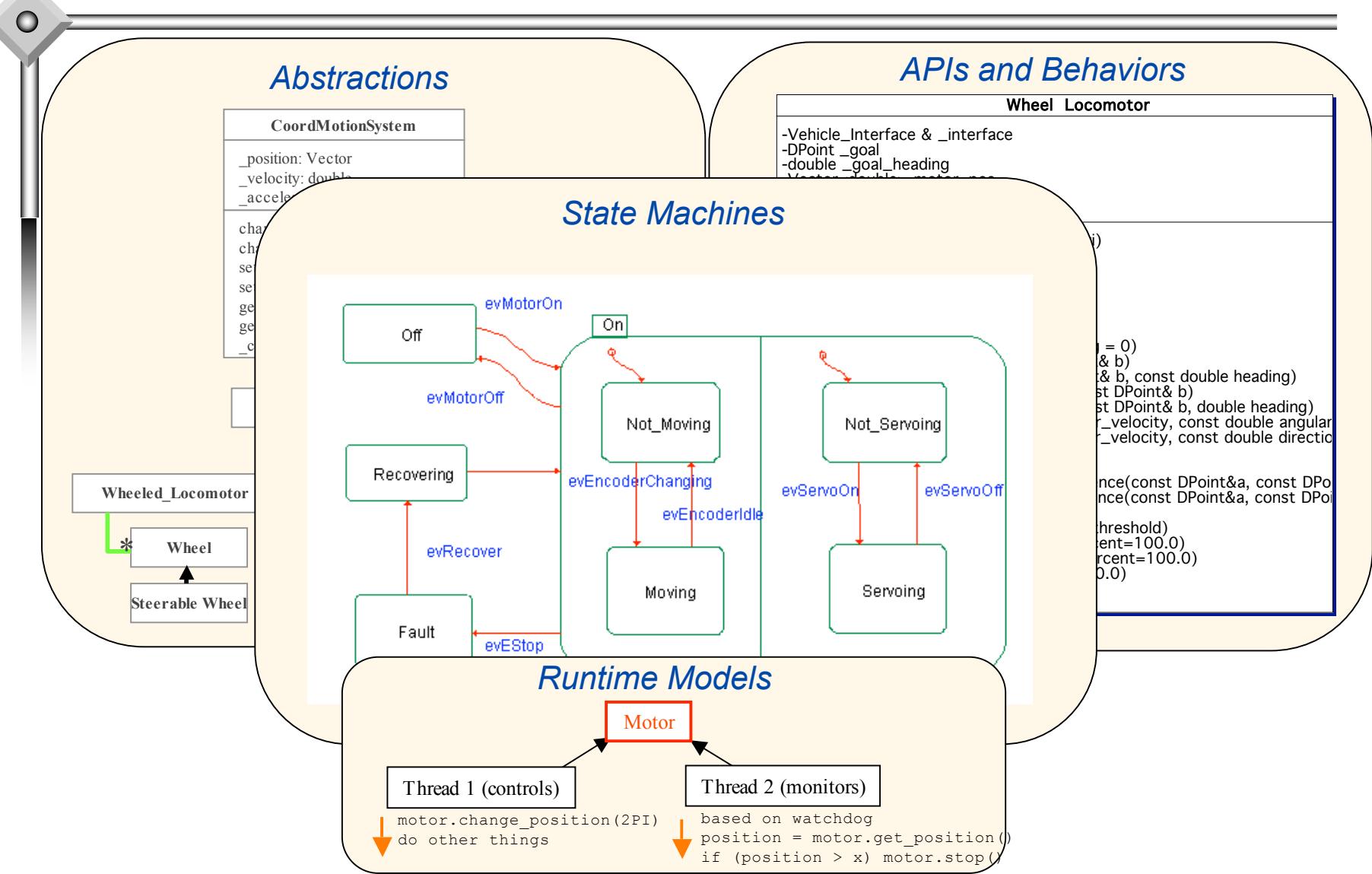


The Functional Layer





Standardizing Base Abstractions

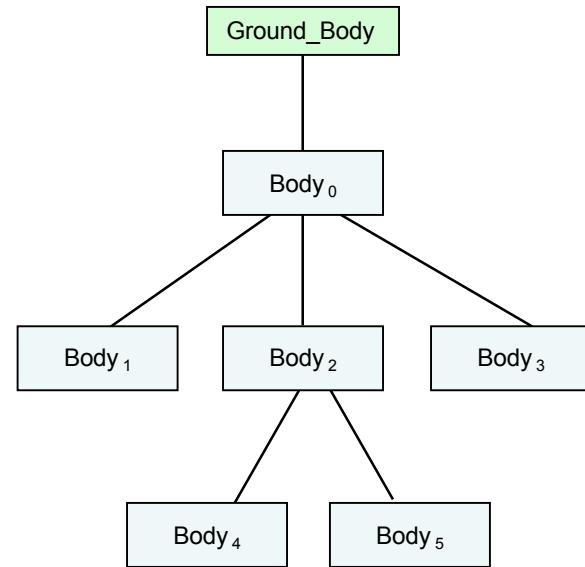




Unified Mechanism Model

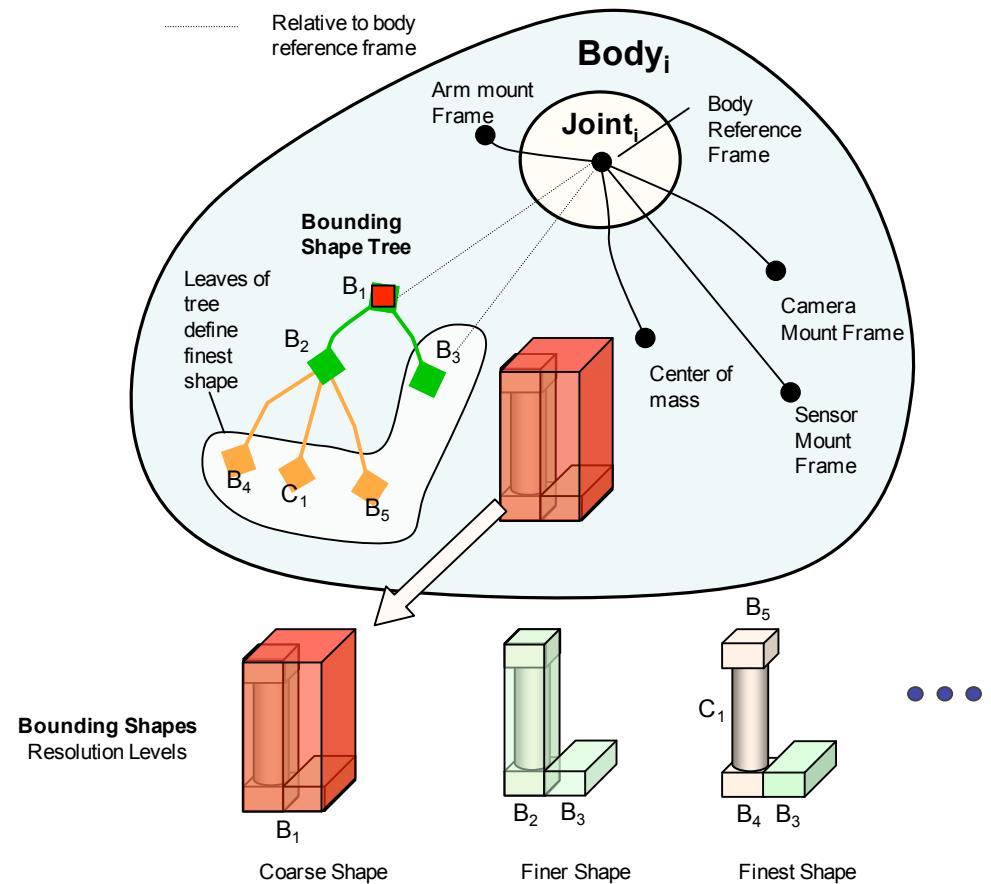


Body Tree



Mechanism Tree

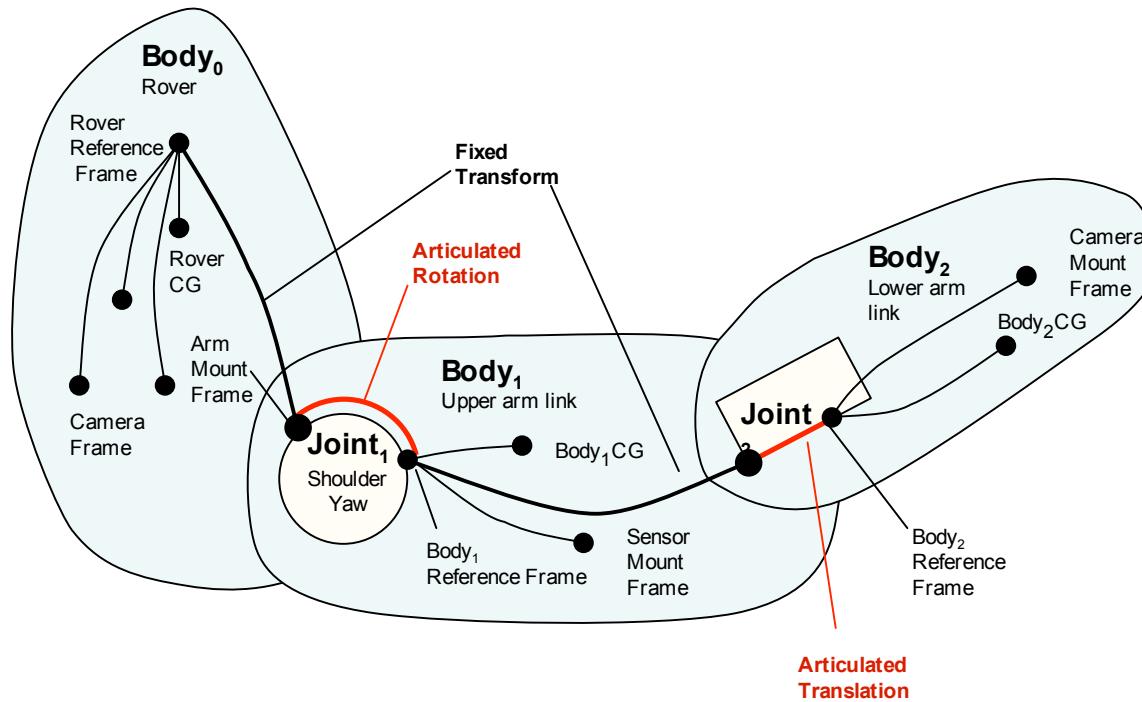
Bodies and Joints





Connecting Bodies and Joints

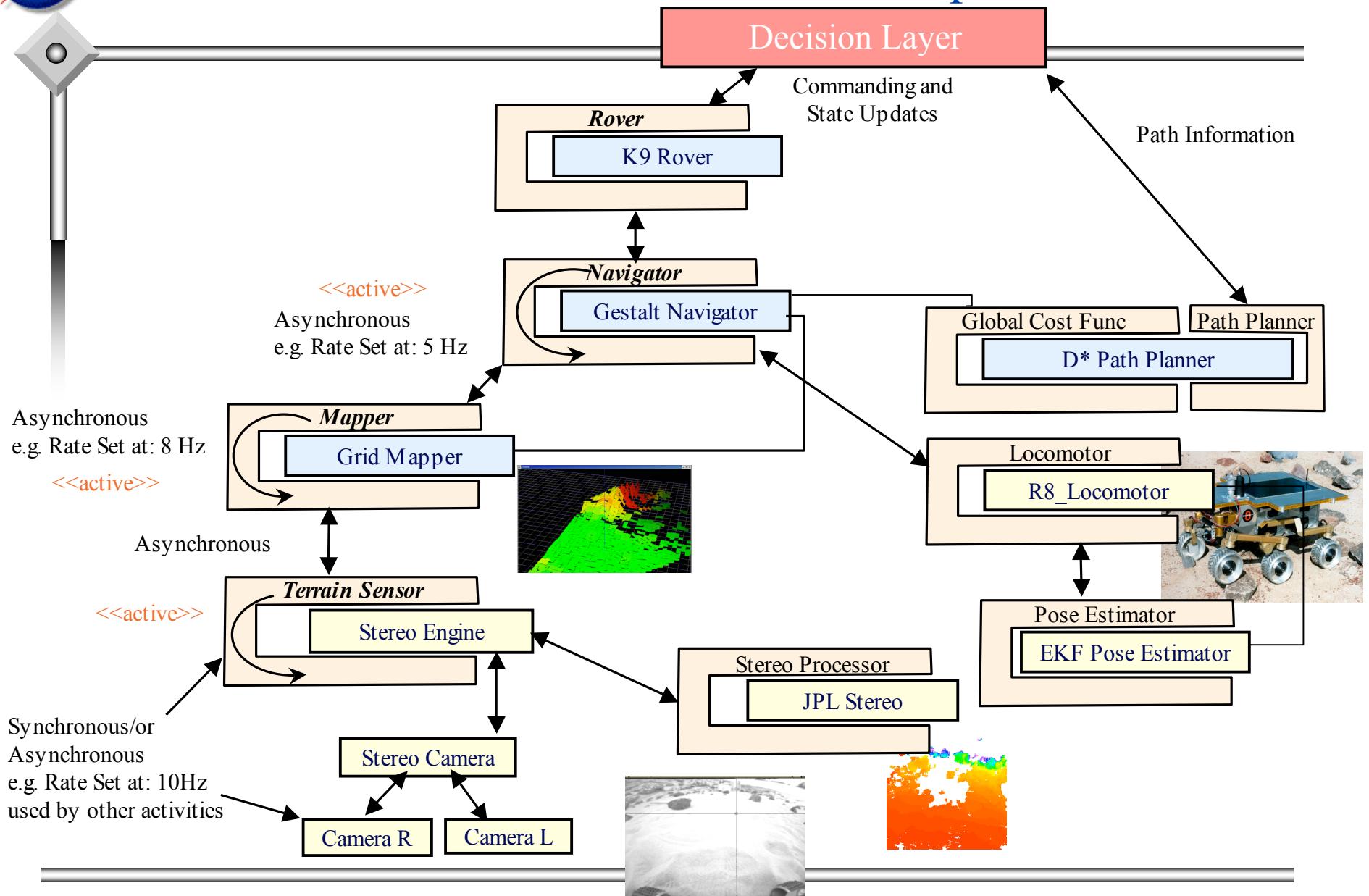
JPL





JPL

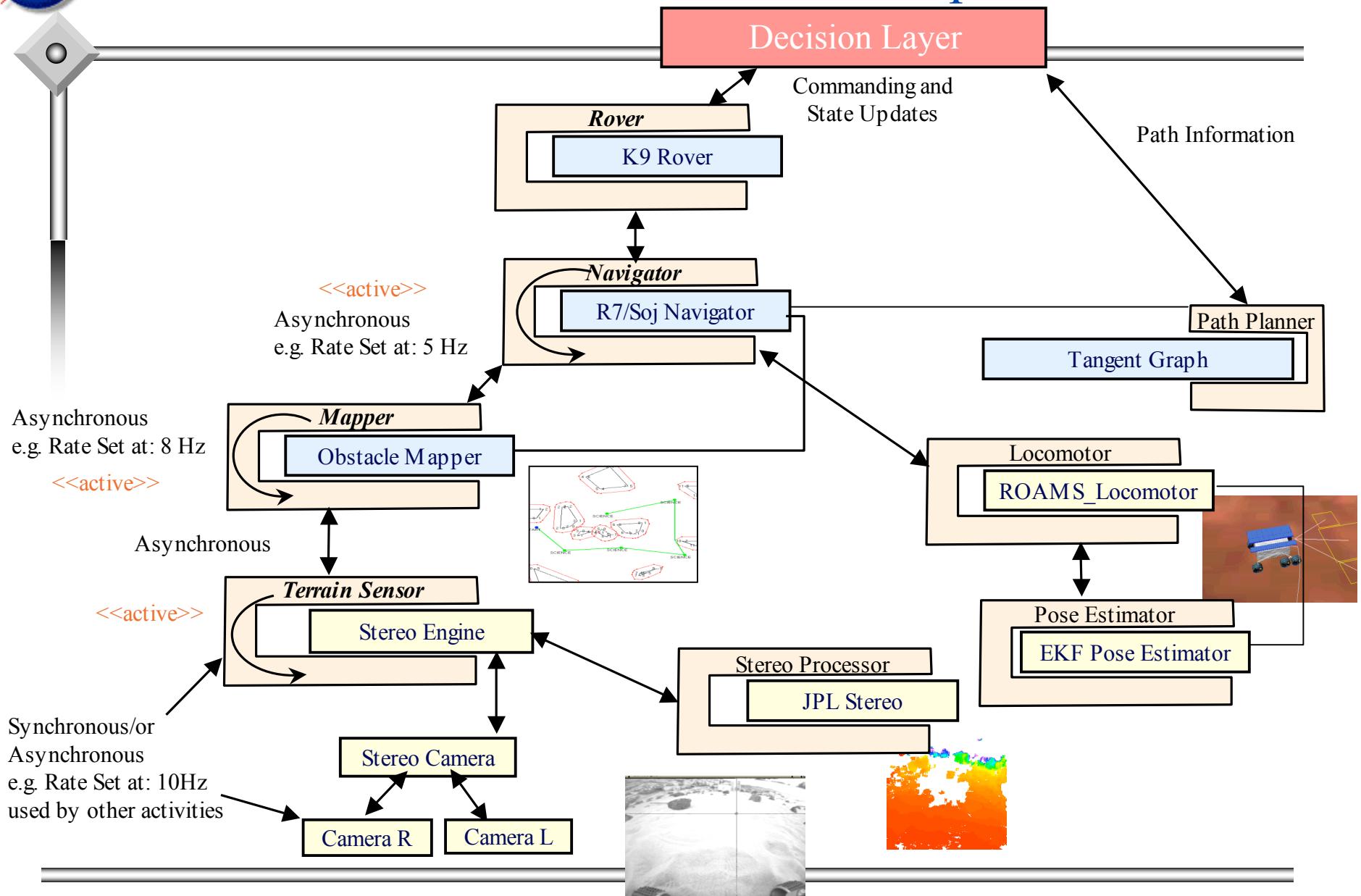
Architectural Traverse Example





JPL

Architectural Traverse Example





Some Results on Reusability





Some Software Inter-operability Statistics



Algorithm Description	1st Adaptation		Subsequent Adaptations (time in days)		
	On	Time	On	Adapt.Time	Tuning Time
3D Extended Kalman Filter for Rover Pose Estimation	Rocky 8	60	FIDO (JPL)	1	3
3D Locomotion for wheeled vehicles	Rocky 7	30	Rocky 8 (JPL)	10	12
			FIDO (JPL)	1	1
			K9 (ARC)	4	1
Morphin Navigator	Rocky 8	75	FIDO (JPL)	1	1
			K9 (ARC)	4	14
			ROAMS (JPL)	1	-
Mast Control Software	Dexter	21	Rocky 8 (JPL)	7	7
			FIDO (JPL)	4	1



Code Reusability for Motion Control

Rocky 7 Modules	Lines of Code	Status
Controlled Motor	2,652	Reusable
Input Output	2,690	Reusable
Bits	1,580	Reusable
Resources (Timers, etc)	725	Reusable
Rocky 7 Motor	927	Non-reusable
Rocky 7 H/W Maps	841	Non-reusable
Motor Controller LM629	1,143	R
Digital I/O Board (S720)	576	R
PCI Components	329	R
Total	11,463	
Total Reusable	85%	
Total Reusable - Strict	67%	

Rocky 8 Modules	Lines of Code	Status
Controlled Motor	2,652	F
Trajectory Generator	691	F
Input Output	2,690	F
Bits	1,580	F
Resources (Timers, etc.)	725	F
Bits	756	F
Rocky 8 Motor	1,180	Non-reusable
Rocky 8 H/W Maps	626	Non-reusable
Widget Board Software	2,126	Reusable - widget
Motor Controller HCTL	900	Reusable - HCTL
I2C Master	1,165	Reusable - I2C
I2C Master Tracii	1,223	Reusable - Tracii
Total	16,314	
Total Reusable	89%	
Total Reusable - Strict	56%	

FIDO Modules	Lines of Code	Status
Controlled Motor	2,652	Reusable
Trajectory Generator	691	Reusable
PID Controller	997	Reusable
Input Output	2,690	Reusable
Bits	1,580	Reusable
Resources (Timers, etc)	725	Reusable
Common Definitions	2,380	Reusable
FIDO Motor	2,086	Non-reusable
FIDO H/W Maps	1,494	Non-reusable
Encoder Counter (ISA P 400)	463	Reusable - H/W
Analog Input Board (MSI P415)	519	Reusable - H/W
Analog Output Board (MSI P460)	462	Reusable - H/W
Digital I/O Board (MSI P560)	602	Reusable - HCTL
Total	17,341	
Total Reusable	79%	
Total Reusable - Strict	68%	



Code Reusability for Locomotion Example

JPL

Module	Lines of Code	Status	Depends On
Wheel Locomotor	1445	Reusable	Motion Sequence, 1D Solver, Homogeneous Transforms
Motion Sequence	540	Reusable	Vector
Matrix, Vector, Array	1083	Reusable	-
1D Solver	356	Reusable	-
Location, Homogeneous Transforms	341	Reusable	Rotation Matrix, Point 2D
Rotation Matrices	435	Reusable	-
Point 2D	131	Reusable	-
Controlled Motor	2080	Reusable	
Rocky 8 Locomotor	250	Non-reusable	Rocky 8 Motor
Rocky 8 Motor	334	Non-reusable	Widget Motor, etc...
Total	6995	584 (non-reusable)	
Total Reusable	~92%		



Conclusions



- Use abstraction to master complexity
- Encapsulate and abstract hardware variations
- Provide multi-level access through Decision Layer for fault diagnosis and recovery
- Use domain expertise to guide design
- Make all assumptions explicit
- Stabilize external interfaces rapidly
- Document processes and products well
- Avoid over-generalization - define scope
- Encapsulate system specific runtime models
- Do not comprise performance - least common denominator solutions are unacceptable in hw/sw interactions
- Standardize Hardware



Acknowledgements

CLARAty Team (multi-center)



Jet Propulsion Laboratory

- ROAMS/Darts Team
- CLEaR Team
- Instrument Simulation Team
- Machine Vision Group
- Robotic Systems Group



Ames Research Center

- K9 Team

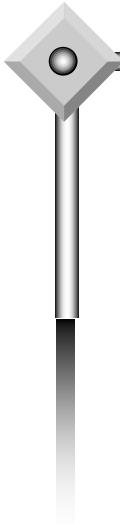


Carnegie Mellon University

University of Minnesota

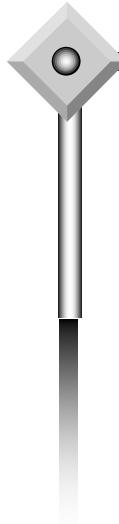


Thank you for your Attention



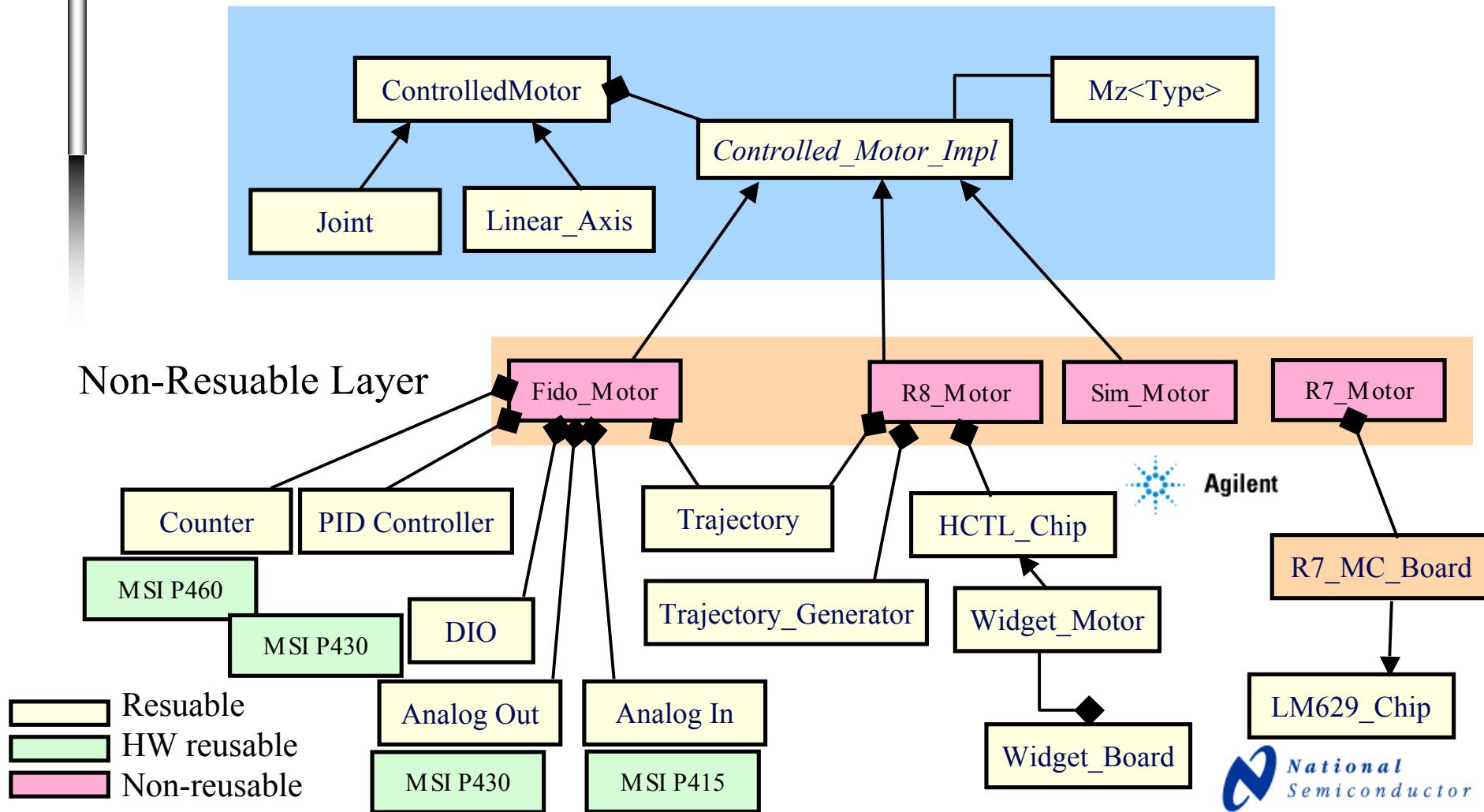


Backup Slides



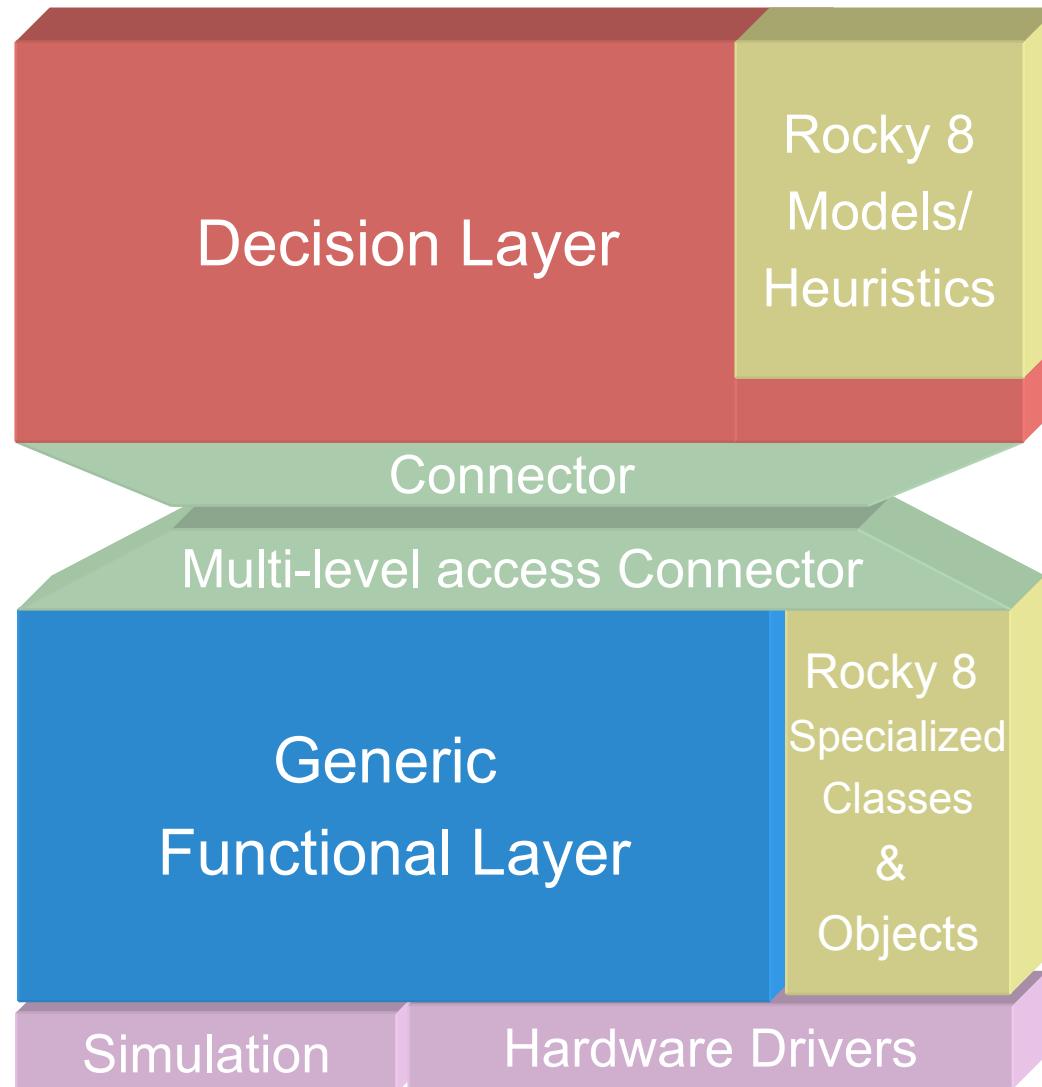


Examples of CLARAty Reusability





Adapting to a Rover





Supported Platforms



Rocky 8

VxWorks x86

JPL



K9

Linux
x86

Ames



Rocky 7

VxWorks ppc

JPL



FIDO

VxWorks x86

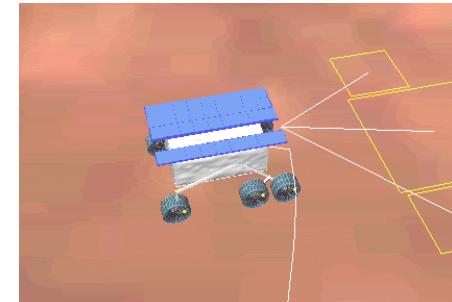
JPL



ATRV

Linux x86

CMU



ROAMS

Solaris Linux

JPL



CLARAty Team



NASA Ames Research Center

- Maria Bualat
- Sal Desiano
- Clay Kunz (*Data Structure Lead*)
- Eric Park
- Randy Sargent
- Anne Wright (*Cog-E & Core lead*)

Carnegie Mellon University

- David Apelfaum
- Reid Simmons (*Navigation lead*)
- Chris Urmson
- David Wettergreen

University of Minnesota

- Stergios Roumeliotis
- Yukikazu Hidaka

Jet Propulsion Laboratory

- Max Bajracharya (34) (*Cog-E & Vision lead*)
- Edward Barlow (34)
- Antonio Diaz Calderon (34)
- Caroline Chouinard (36)
- Gene Chalfant (34)
- Tara Estlin (36) (*Deputy Manager & Decision Layer lead*)
- Erann Gat (36)
- Dan Gaines (36) (*Estimation Lead*)
- Mehran Gangianpour (34)
- Won Soo Kim (34) (*Motion lead*)
- Michael Mossey (31)
- Issa A.D. Nesnas (34) (*Task Manager*)
- Richard Petras (34) (*Adaptation lead*)
- Marsette Vona (34)
- Barry Werger (34)

OphirTech

- Hari Das Nayar

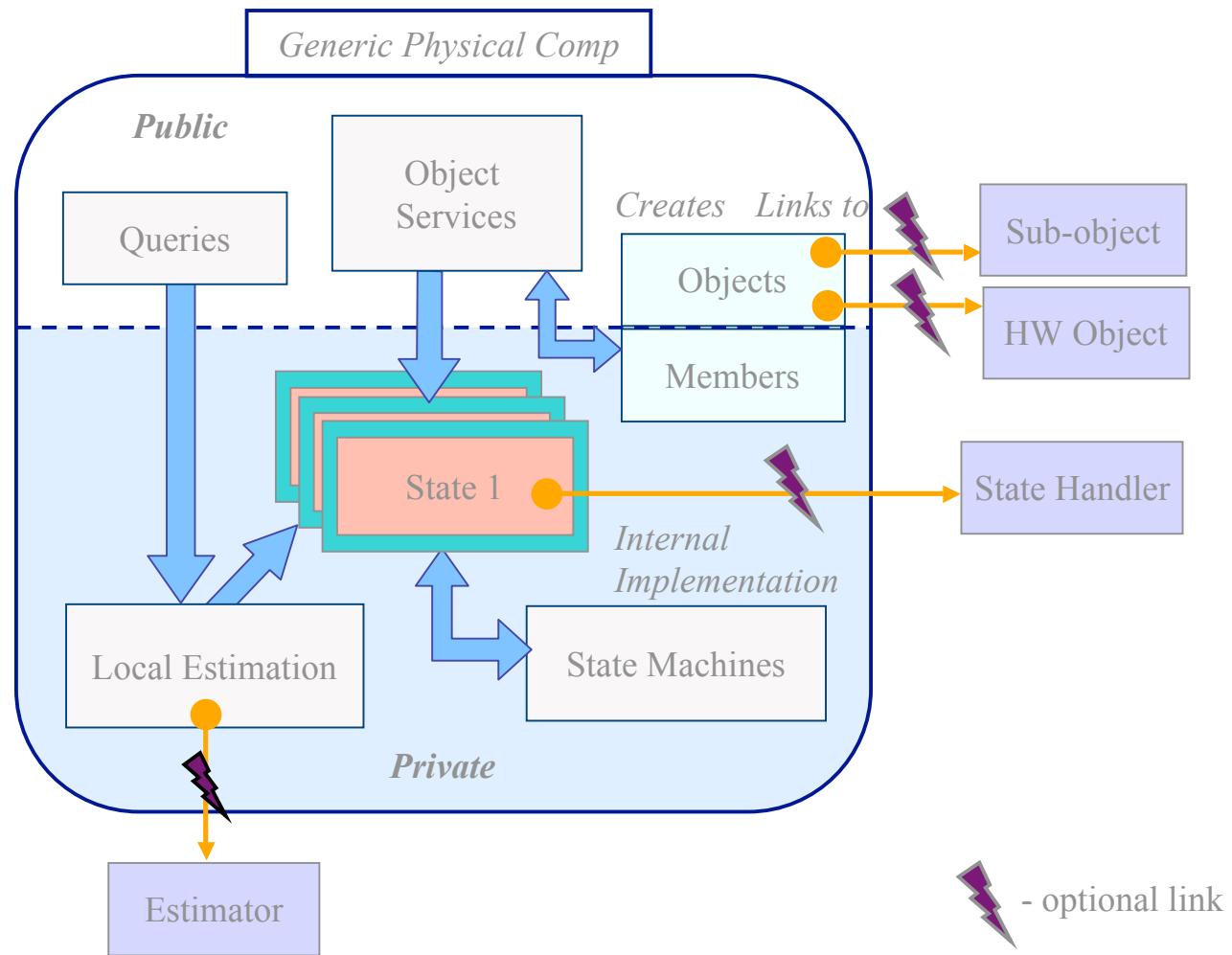


Summary



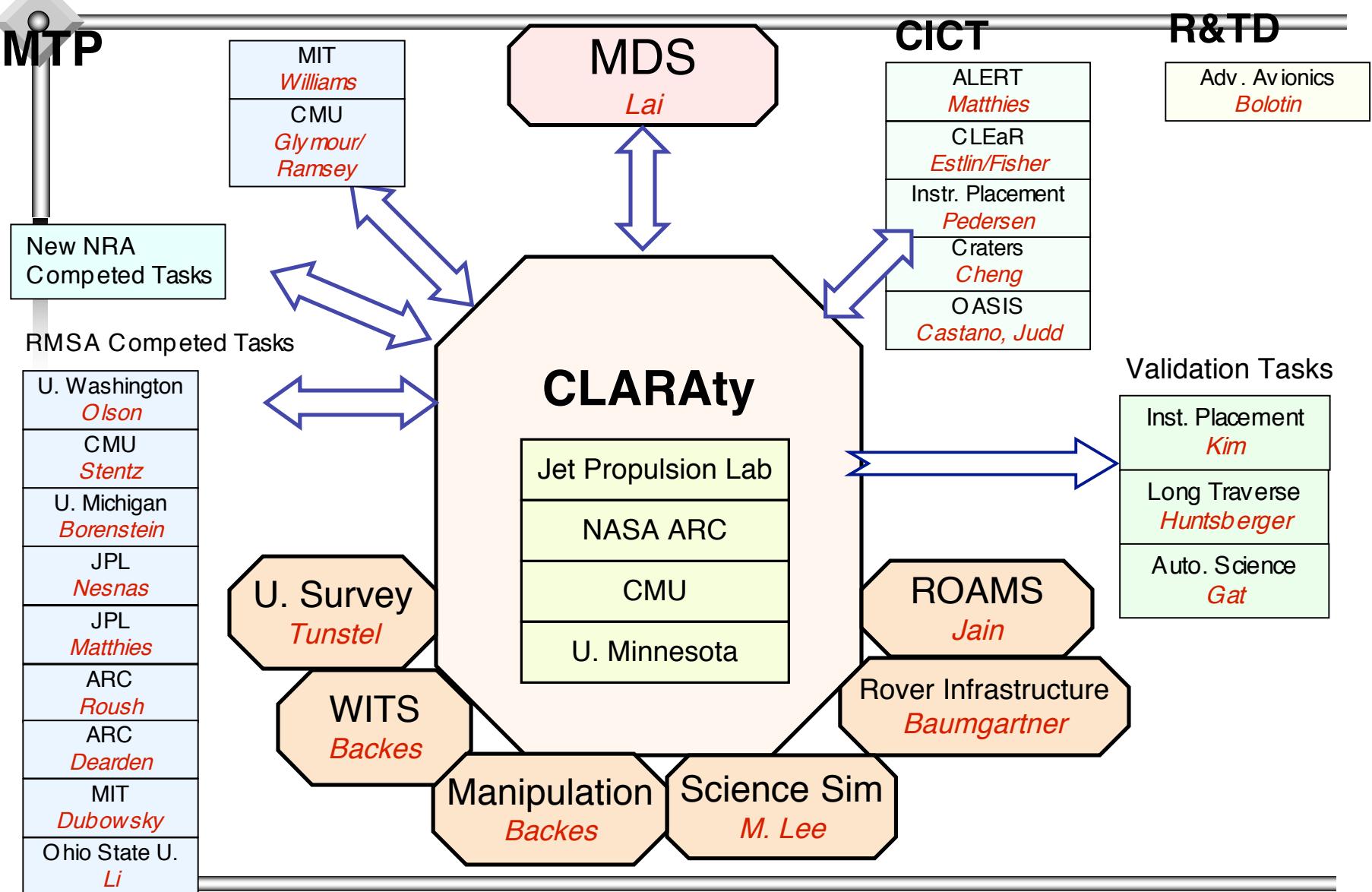
- CLARAty provides a repository of reusable software components at various abstraction levels
- It attempts at capturing well-known robot technologies in a basic framework for researchers
- It publishes the behavior and interfaces of its components
- It allows researchers to integrate novel technologies at different levels of the architecture
- It is a collaborative effort within the robotics community
- It will run on multiple heterogeneous robots

Component Analysis



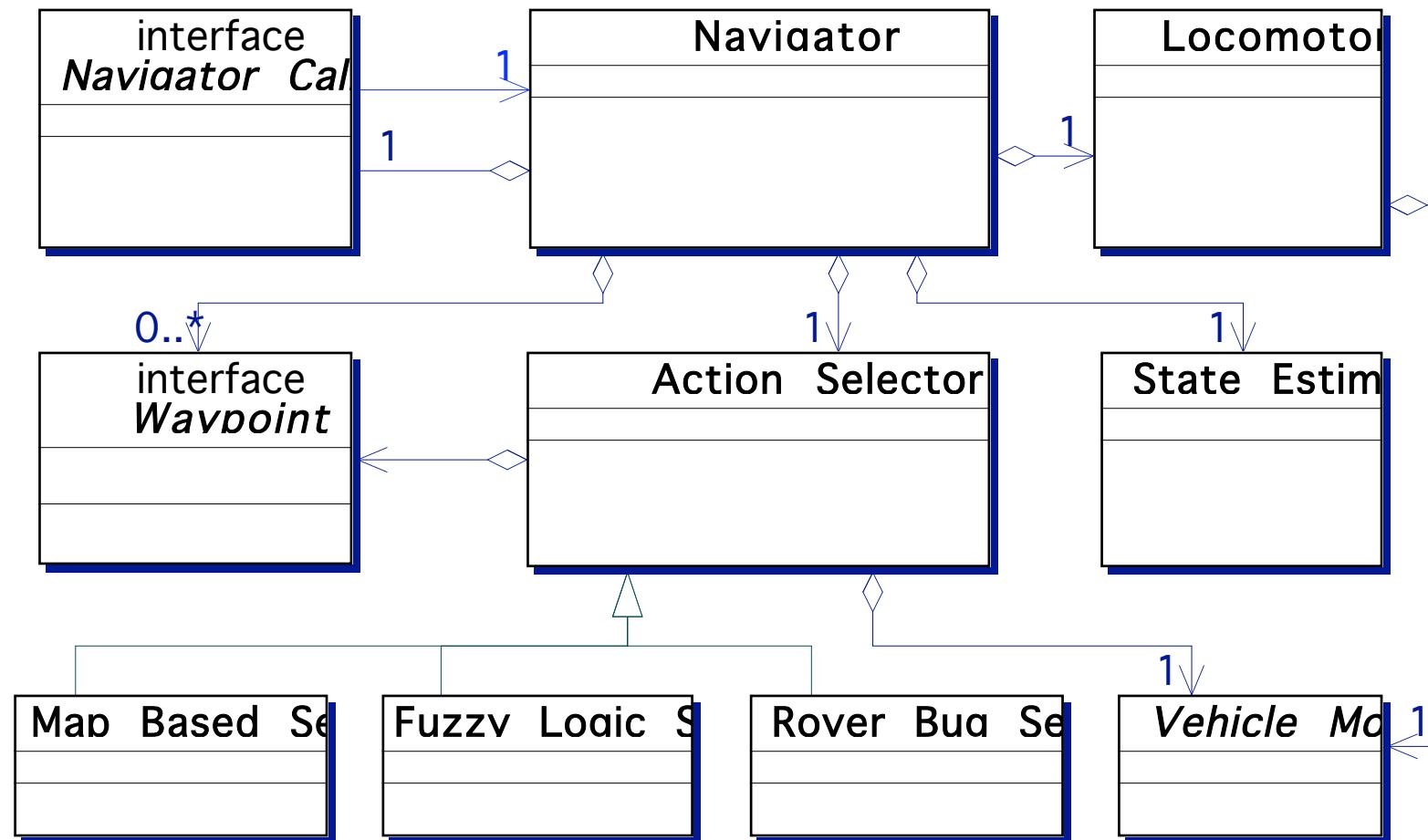


Collaborations



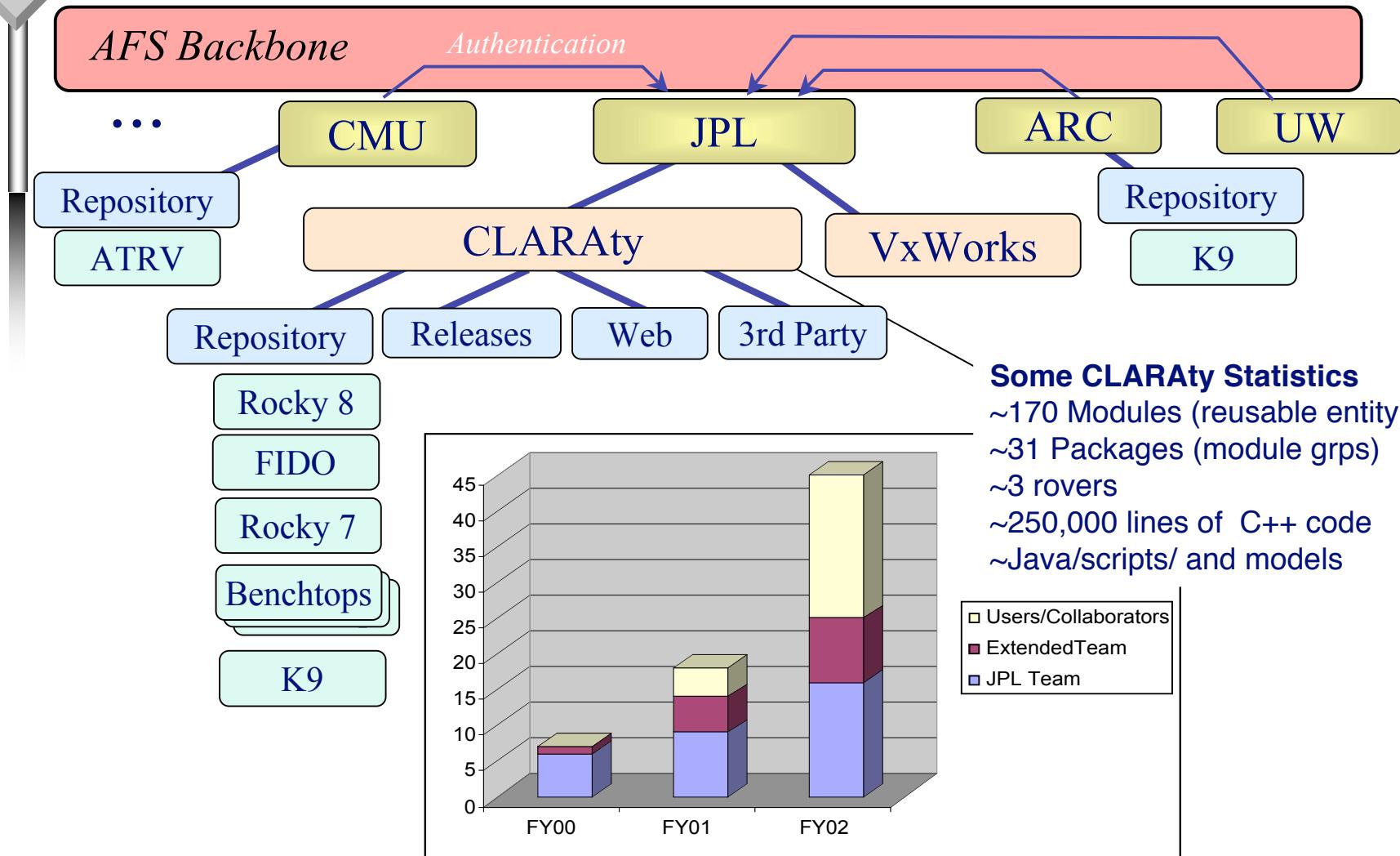


JPL



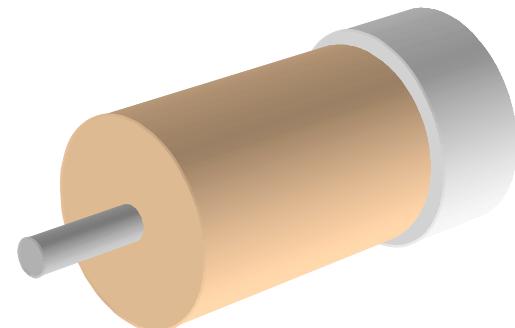
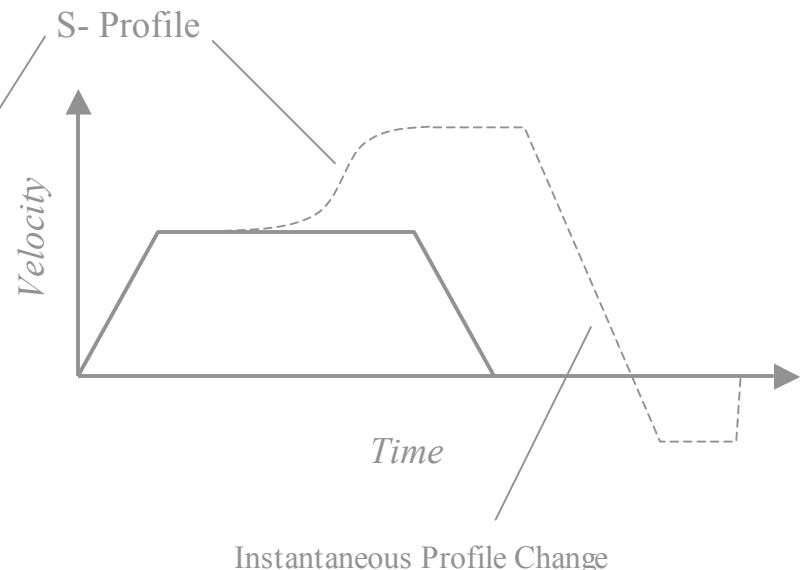


Software Development Process



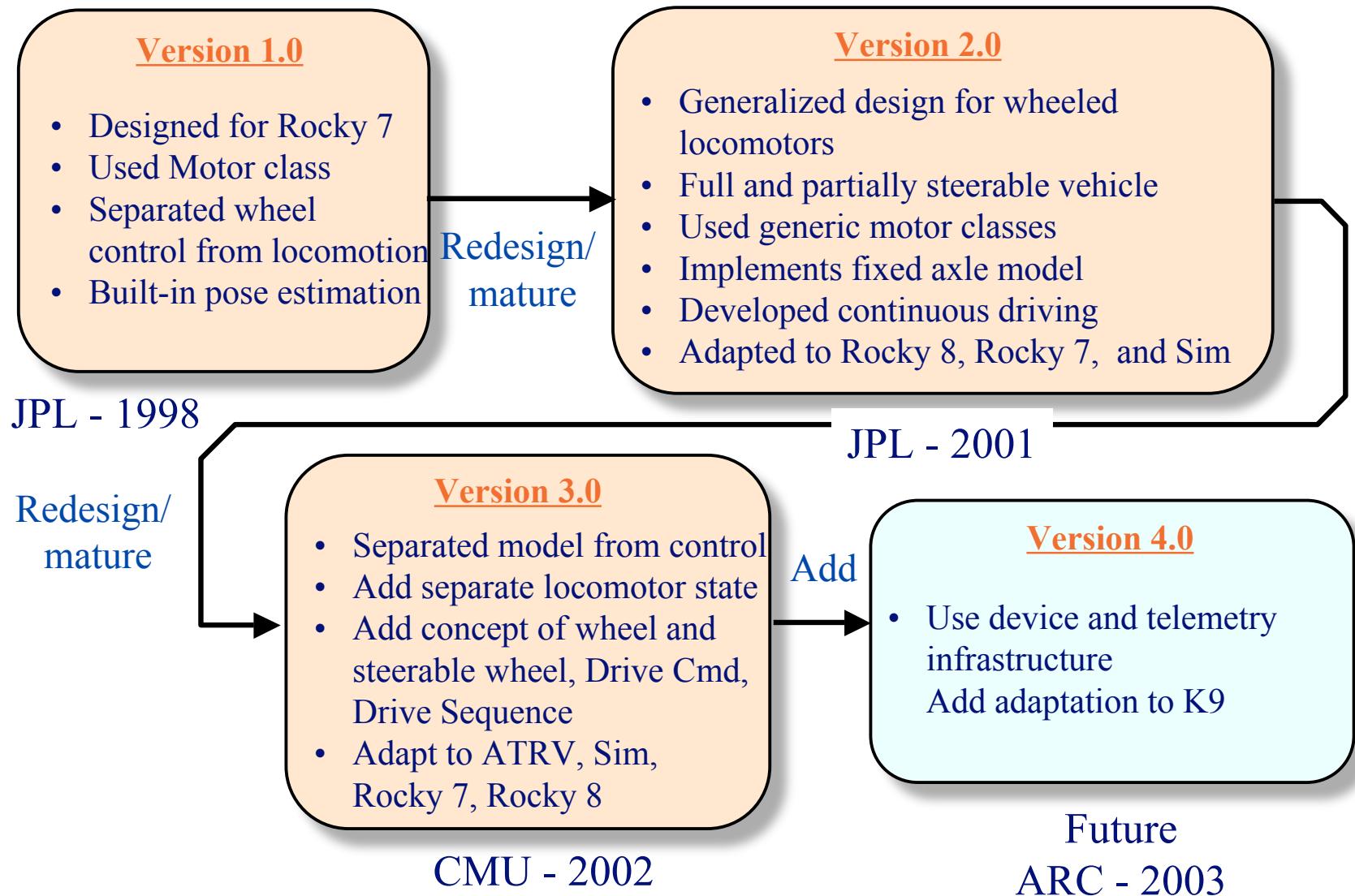
Example: Generic Controlled Motor

- Define generic capabilities independent of hardware
- Provide implementation for generic interfaces to the best capabilities of hardware
- Provide software simulation where hardware support is lacking
- Adapt functionality and interface to particular hardware by specialization inheritance
- Motor Example: public interface command groups:
 - Initialization and Setup
 - Motion and Trajectory
 - Queries
 - Monitors & Diagnostics



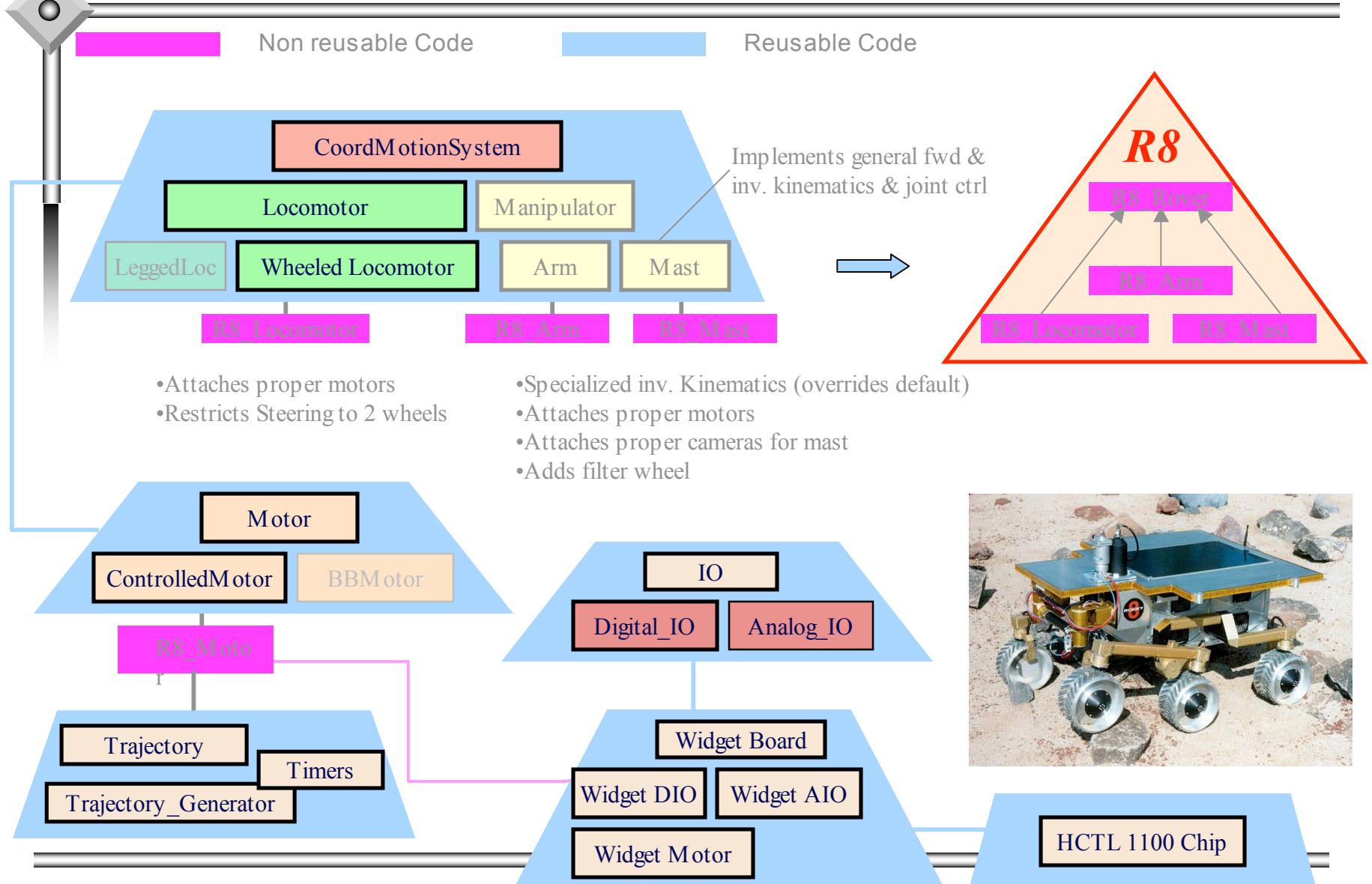


Example: collaborative development for locomotor





R8 Specific Rover Implementation





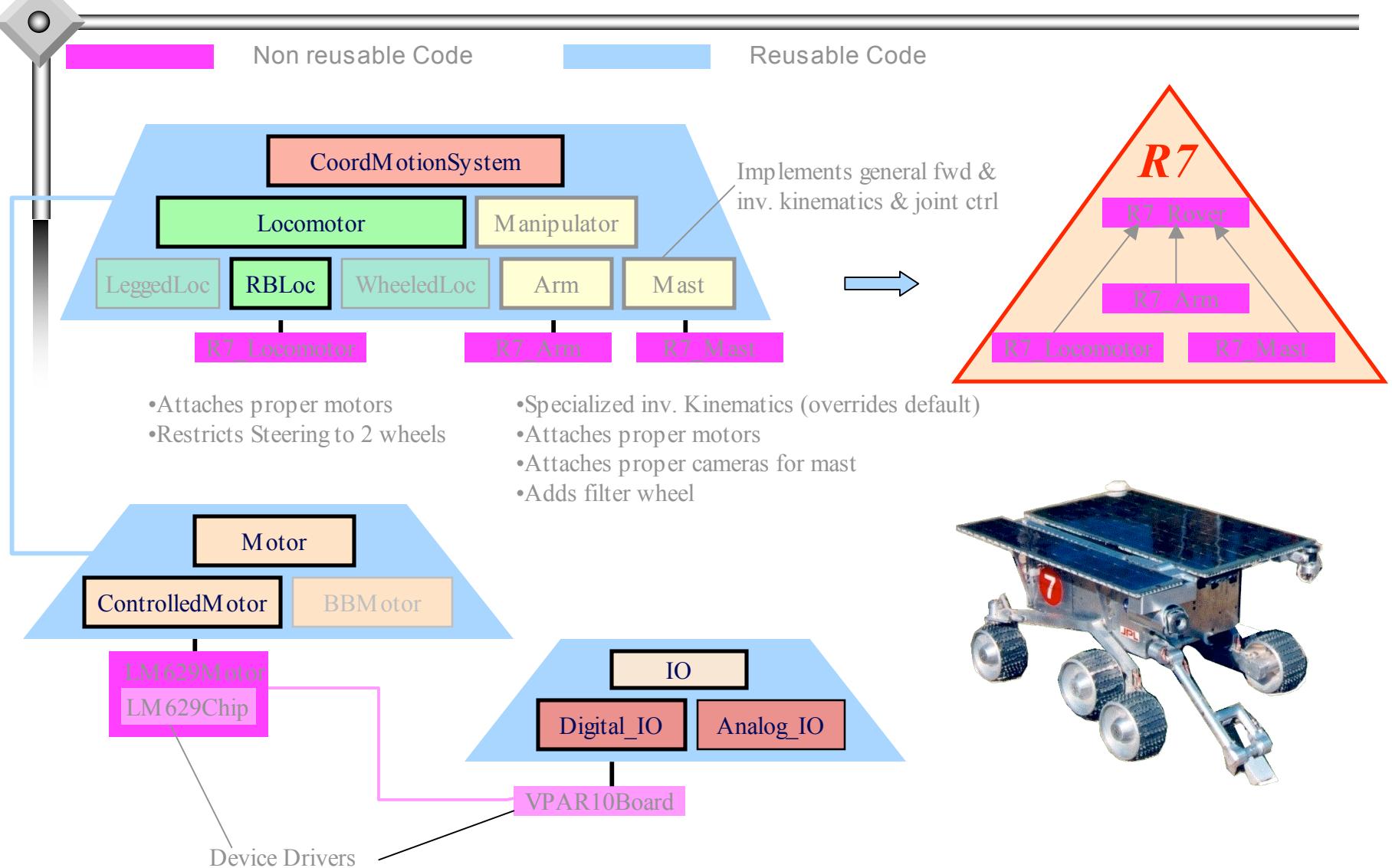
Capabilities of Wheel Locomotor



- Type of maneuvers:
 - Straight line motions (fwd / bkwd)
 - Crab maneuvers
 - Arc maneuvers
 - Arc crab maneuvers
 - Rotate-in-place maneuvers (arc turn $r=0$)
- Driving Operation
 - Non-blocking drive commands
 - Multi-threaded access to the `Wheel_Locomotor` class – e.g. one task can use `Wheel_Locomotor` for driving while the other for position queries
 - Querying capabilities during all modes of operation. Examples include position updates and state queries
 - Built-in rudimentary pose estimation that assumes vehicle follows commanded motion



R7 Specific Rover Implementation





Why is robotic software “hard”?



- Software:
 - Software is large and complex
 - Has lots of diverse functionality
 - Integrates many disciplines
 - Requires real-time runtime performance
 - Talks to hardware
- Hardware:
 - Physical and mechanics are different
 - Electrical hardware architecture changes
 - Hardware component capabilities vary



What is CLARAty?



CLARAty is a *unified* and *reusable* software that provides robotic functionality and simplifies the integration of new technologies on robotic platforms

A research tool for technology development and maturation